**Spécifications techniques**

[Nom du projet + nom du client]

|  |  |  |  |
| --- | --- | --- | --- |
| **Version** | **Auteur** | **Date** | **Approbation** |
| 1.0 | DOS SANTOS | 25/06/2024 | [nom de la personne qui doit valider le document] |

[I. Choix technologiques](#_409jm82qlgst) 2

[II. Liens avec le back-end](#_ex0md8td8283) 6

[III. Préconisations concernant le domaine et l’hébergement](#_a61v3uh5vs19) 6

[IV. Accessibilité](#_n56najuvb5ys) 7

[V. Recommandations en termes de sécurité](#_l80x4g9stizp) 8

[VI. Maintenance du site et futures mises à jour](#_vbbknpfyus5o) 10

## Choix technologiques

* État des lieux des besoins fonctionnels et de leurs solutions techniques :

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Besoin** | **Contraintes** | **Solution** | **Description de la solution** | **Justification (2 arguments)** |
| **Page login :**  Création d'une page de connexion pour les utilisateurs. | La page de connexion doit s'ouvrir sous forme de modale.  L'utilisateur doit pouvoir entrer son adresse e-mail.  Un e-mail de confirmation doit être envoyé pour l'authentification.  Un lien "Besoin d'aide" doit permettre d'envoyer un e-mail à l'équipe de support. | Librairie: react-modal | Cette librairie React permet de créer simplement des modales performantes, accessibles avec un minimum de code. | 1/ Nous avons choisi de développer en React, donc cette librairie est cohérente avec ce choix.  2/ Il s'agit de la librairie la plus utilisée et elle offre une grande flexibilité pour la gestion des modales. |
| **Sécurité et authentification :**  Mettre en place un système de sécurité et d'authentification pour les utilisateurs. | Implémenter HTTPS pour sécuriser les communications.  Mettre en place un système de hashage des mots de passe avec bcrypt.  Utiliser JWT (JSON Web Tokens) pour la gestion des sessions.  Implémenter un rate limiter pour limiter les tentatives de connexion. | Utilisation de bcrypt pour le hashage des mots de passe.  Utilisation de jsonwebtoken pour la gestion des sessions avec JWT.  Utilisation de express-rate-limit pour limiter les tentatives de connexion. | bcrypt permet de hasher les mots de passe de manière sécurisée et est bien supporté par Node.js.  jsonwebtoken permet de créer et vérifier des tokens JWT facilement, assurant une gestion sécurisée des sessions.  express-rate-limit est un middleware pour Express qui permet de limiter les requêtes à une API, protégeant contre les attaques par force brute. | 1/ Ces librairies sont largement utilisées et reconnues pour leur sécurité et leur efficacité.  2/ Elles sont compatibles avec notre stack technologique (Node.js et Express), facilitant leur intégration. |
| **Création de plat :**  Permettre aux restaurateurs d'ajouter des plats à leur menu. | Les utilisateurs doivent pouvoir sélectionner une catégorie de plat.  Une modale doit s'ouvrir pour saisir les détails du plat (photo, nom, prix, description).  Il doit être possible de créer autant de plats que souhaité. | Librairie: react-modal | Cette librairie React permet de créer des modales facilement, ce qui est nécessaire pour la saisie des détails des plats. | 1/ La cohérence avec notre choix de développement en React.  2/ La simplicité et l'efficacité de react-modal pour gérer des modales complexes. |
| **Exportation PDF :**  Permettre aux restaurateurs d'exporter leur menu en PDF. | Le restaurateur doit pouvoir télécharger le fichier PDF du menu en un clic. | Librairie : jspdf | jspdf est une librairie JavaScript permettant de générer des fichiers PDF directement depuis le front-end. | jspdf est bien documentée et largement utilisée pour générer des PDF dans des applications web.  Elle permet une génération rapide et efficace des PDF sans nécessiter de backend complexe pour cette fonctionnalité. |
| **Gestion des réservations :**  Permettre aux restaurateurs de gérer les réservations de table directement depuis MenuMaker. | L'utilisateur doit pouvoir voir et modifier les réservations en cours.  Les clients doivent pouvoir réserver une table via un formulaire sur le site web du restaurateur.  Les notifications de nouvelles réservations doivent être envoyées par e-mail. | Librairie : react-calendar pour la gestion des réservations. | react-calendar permet de créer facilement des calendriers interactifs pour la gestion des réservations. | 1/ La bibliothèque est compatible avec React et permet une intégration facile et rapide.  2/ Elle offre une interface utilisateur conviviale pour la gestion des réservations. |
| **Gestion des pages d'erreur :**  Créer des pages d'erreur pour informer les utilisateurs en cas de problème et améliorer l'expérience utilisateur. | Une page d'erreur 404 (page non trouvée) doit être affichée lorsqu'une URL invalide est saisie.  Une page d'erreur 500 (erreur serveur) doit être affichée en cas de problème interne.  Les pages d'erreur doivent être personnalisées pour correspondre à la charte graphique de l'application.  Les pages doivent offrir des liens de navigation pour retourner à la page d'accueil ou contacter le support. | Librairie : react-router, styled-components ou modules css  **Page 404**  Message : "Page non trouvée"  Description : "La page que vous recherchez n'existe pas."  Actions : Lien pour retourner à la page d'accueil, lien pour contacter le support  **Page 500**  Message : "Erreur interne du serveur"  Description : "Une erreur est survenue sur le serveur. Veuillez réessayer plus tard."  Actions : Lien pour retourner à la page d'accueil, lien pour contacter le support | react-router permet de gérer les routes de l'application et de rediriger vers des pages d'erreur spécifiques en cas de route non valide.  styled-components ou modules css permet de styliser les pages d'erreur conformément à la charte graphique de l'application. | 1/ react-router est la bibliothèque standard pour la gestion des routes dans les applications React, assurant une redirection efficace vers les pages d'erreur.  2/ styled-components et modules css permettent de maintenir une cohérence visuelle avec le reste de l'application, améliorant ainsi l'expérience utilisateur même en cas d'erreur. |

## Liens avec le back-end

* Quel langage pour le serveur ? Node.js
* A-t-on besoin d’une API ? Oui, RESTful API avec Express.js
* Base de données choisie : MongoDB (NoSQL)

## Préconisations concernant le domaine et l’hébergement

**Nom du domaine :**

**Domaine principal :** qwenta.fr

Sous-domaines possibles :

app.qwenta.fr (pour l'application web)

api.qwenta.fr (pour les services API)

support.qwenta.fr (pour le support et l'assistance)

Nom de l’hébergement :

AWS (Amazon Web Services)

Adresses e-mail :  
contact@qwenta.fr : Pour les demandes générales.

support@qwenta.fr : Pour l'assistance technique et le support client.

no-reply@qwenta.fr : Pour les e-mails transactionnels (confirmation de compte, réinitialisation de mot de passe).

## Accessibilité

Compatibilité navigateur :

Pour garantir une accessibilité optimale du site Qwenta.fr, il est essentiel d'assurer la compatibilité avec les navigateurs suivants :

Navigateurs Desktop :

* Google Chrome (dernière version)
* Mozilla Firefox (dernière version)
* Apple Safari (dernière version)

Types d’appareils :

Le site doit être accessible et fonctionnel sur une variété d'appareils pour garantir une expérience utilisateur optimale, quel que soit le dispositif utilisé. Les types d'appareils à considérer sont :

Ordinateurs de bureau et portables :

* Résolution minimale : 1024x768 pixels
* Résolution recommandée : 1366x768 pixels et plus

## Recommandations en termes de sécurité

**Accès aux comptes**

Pour garantir la sécurité des utilisateurs et des données sur le site Qwenta.fr, il est crucial de mettre en place des mesures de sécurité robustes pour l'accès aux comptes.

1. **Authentification et autorisation :**

* HTTPS : Utiliser HTTPS pour toutes les communications entre le client et le serveur afin de chiffrer les données en transit.
* Hashage des mots de passe : Stocker les mots de passe hachés en utilisant des algorithmes sécurisés comme bcrypt.
* Vérification des e-mails : Envoyer des e-mails de confirmation lors de la création de compte pour vérifier l'identité des utilisateurs.
* Authentification à deux facteurs (2FA) : Offrir une option d'authentification à deux facteurs pour une sécurité renforcée.
* Gestion des sessions : Utiliser des jetons JWT (JSON Web Tokens) pour la gestion des sessions avec une expiration et une invalidation appropriée des jetons.
* Rate limiting : Implémenter un système de limitation des taux pour les tentatives de connexion pour prévenir les attaques par force brute.

2. **Gestion des autorisations :**

* Contrôle d'accès basé sur les rôles (RBAC, Role-Based Access Control) : Mettre en place des contrôles d'accès basés sur les rôles pour limiter l'accès aux fonctionnalités en fonction des droits des utilisateurs.
* Revue régulière des permissions : Effectuer des revues régulières des permissions des utilisateurs pour s'assurer qu'ils disposent des accès appropriés.

**Librairies et dépendances**

L'utilisation de librairies et de dépendances tierces peut introduire des vulnérabilités si elles ne sont pas correctement gérées.

1. **Choix des librairies :**

* Sources fiables : Utiliser uniquement des librairies et des dépendances provenant de sources fiables et bien maintenues.
* Popularité et maintenance : Choisir des librairies populaires avec une communauté active et un cycle de mise à jour régulier.

2. **Gestion des dépendances :**

* Audit de sécurité : Effectuer des audits de sécurité réguliers sur les dépendances en utilisant des outils comme npm audit pour identifier et corriger les vulnérabilités.
* Mises à jour régulières : Garder les dépendances à jour pour bénéficier des correctifs de sécurité et des améliorations.
* Politique de version : Utiliser des versions spécifiques (semver) pour éviter les mises à jour non contrôlées qui pourraient introduire des vulnérabilités ou des incompatibilités.

3. **Isoler les plugins :**

* Sandboxing : Isoler les librairies dans des environnements de sandbox pour limiter leur accès aux ressources sensibles du système. Utiliser Docker pour créer des conteneurs isolés où les librairies peuvent être exécutées en toute sécurité. Par exemple, chaque librairie ou groupe de librairies peut être placé dans un conteneur Docker, limitant ainsi leur accès aux ressources système principales.
* Revues de code : Effectuer des revues de code régulières sur les intégrations de librairies pour s'assurer qu'ils respectent les normes de sécurité de l'application.

## Maintenance du site et futures mises à jour

#### Grandes lignes du contrat de maintenance

Un contrat de maintenance est essentiel pour assurer la performance, la sécurité et la continuité du service de Qwenta.fr. Voici les grandes lignes proposées pour ce contrat de maintenance :

1. **Objet du contrat**

Définir clairement les services couverts par le contrat de maintenance, y compris la gestion des incidents, les mises à jour logicielles, la sécurité, la performance et le support utilisateur.

1. **Durée du contrat**

Spécifier la durée du contrat de maintenance, avec des options de renouvellement.

1. **Niveaux de service (SLA, Service Level Agreement)**

**Disponibilité :** Garantir un temps de disponibilité (uptime) minimum, par exemple, 99.9%.

**Temps de réponse :** Définir des temps de réponse spécifiques pour les différentes catégories d'incidents (critique, majeur, mineur).

**Temps de résolution :** Définir les délais pour résoudre les incidents en fonction de leur gravité.

1. **Maintenance préventive**

**Mises à jour logicielles :** Planifier et exécuter régulièrement des mises à jour pour le système d'exploitation, les serveurs web, les bases de données, et les dépendances de l'application.

**Audits de sécurité :** Effectuer des audits de sécurité réguliers pour identifier et corriger les vulnérabilités potentielles.

**Sauvegardes :** Mettre en place des politiques de sauvegarde régulières, avec des tests de restauration pour s'assurer de l'intégrité des données.

1. **Maintenance corrective**

**Gestion des incidents :** Procédures pour la détection, la classification et la résolution des incidents techniques.

**Corrections de bugs :** Prioriser et corriger les bugs signalés par les utilisateurs ou détectés lors des tests.

1. **Mises à jour évolutives**

**Nouvelles fonctionnalités :** Planification et intégration des nouvelles fonctionnalités basées sur les retours des utilisateurs et les besoins commerciaux.

**Améliorations de performance :** Optimisation continue du code et des infrastructures pour améliorer la performance et l'expérience utilisateur.

1. **Support utilisateur**

**Canaux de support :** Proposer plusieurs canaux de support (e-mail, chat en direct, téléphone) pour assister les utilisateurs.

**Horaires de support :** Définir les heures de disponibilité du support (par exemple, 9h-18h, du lundi au vendredi) et les options pour le support d'urgence en dehors des heures normales.

1. **Rapports et suivis**

**Rapports réguliers :** Fournir des rapports mensuels ou trimestriels sur l’état du site, les incidents résolus, les mises à jour effectuées et les recommandations pour les améliorations futures.

**Réunions de suivi :** Organiser des réunions régulières avec le client pour discuter des performances, des problèmes rencontrés et des priorités pour les futures mises à jour.

1. **Confidentialité et sécurité des données**

**Protection des données :** Assurer la conformité avec les réglementations en matière de protection des données (par exemple, RGPD).

**Accès sécurisé :** Utiliser des pratiques de sécurité robustes pour l'accès aux systèmes et aux données, incluant l'authentification forte et la gestion des accès.

1. **Clause de révision**

**Révision du contrat :** Prévoir des clauses pour la révision périodique du contrat afin de l'adapter aux évolutions technologiques et aux besoins de l'entreprise.